Autonomous Navigation for Flying Robots

Lecture 3.1:
3D Geometry

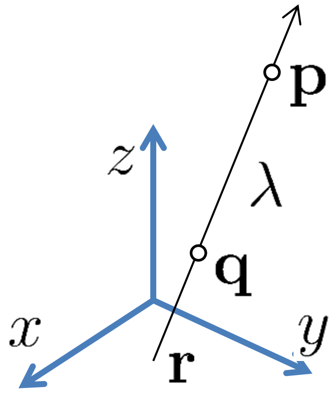
Jürgen Sturm
Technische Universität München

Points in 3D

- 3D point
$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$$
- Augmented vector
$$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4$$
- Homogeneous coordinates
$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^3$$

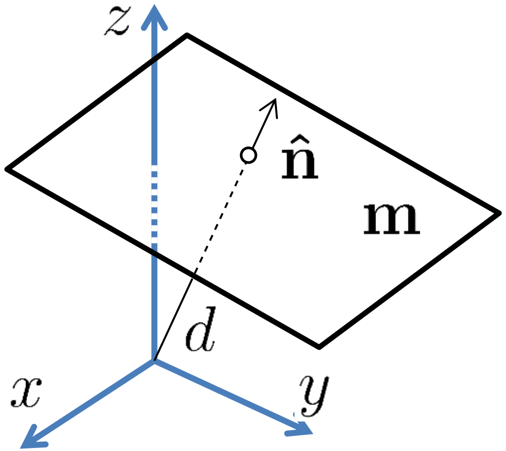
Geometric Primitives in 3D

- 3D line $\mathbf{r} = (1-\lambda)\mathbf{p} + \lambda\mathbf{q}$ through points $\mathbf{p}, \mathbf{q}$
- Infinite line: $\lambda \in \mathbb{R}$
- Line segment joining $\mathbf{p}, \mathbf{q}$: $0 \leq \lambda \leq 1$



Geometric Primitives in 3D

- 3D plane $\tilde{\mathbf{m}} = (a, b, c, d)^\top$
- 3D plane equation $\bar{\mathbf{x}} \cdot \tilde{\mathbf{m}} = ax + by + cz + d = 0$
- Normalized plane $\mathbf{m} = (\hat{n}_x, \hat{n}_y, \hat{n}_z, d)^\top = (\hat{\mathbf{n}}, d)$
  with unit normal vector $\|\hat{\mathbf{n}}\| = 1$ and distance d



3D Transformations

- Translation

$$\tilde{\mathbf{x}}' = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}}_{4\times 4} \tilde{\mathbf{x}}$$

- Euclidean transform (translation + rotation), (also called the Special Euclidean group SE(3))

$$\tilde{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{x}}$$

- Scaled rotation, affine transform, projective transform…

3D Euclidean Transformations

- Translation $\mathbf{t} \in \mathbb{R}^3$ has 3 degrees of freedom
- Rotation $\mathbf{R} \in \mathbb{R}^{3\times 3}$ has 3 degrees of freedom

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D Rotations

- A rotation matrix is a 3x3 orthogonal matrix

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

- Also called the special orientation group SO(3)
- Column vectors correspond to coordinate axes

3D Rotations

- What operations do we typically do with rotation matrices?
  - Invert, concatenate
  - Estimate/optimize
- How easy are these operations on matrices?

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

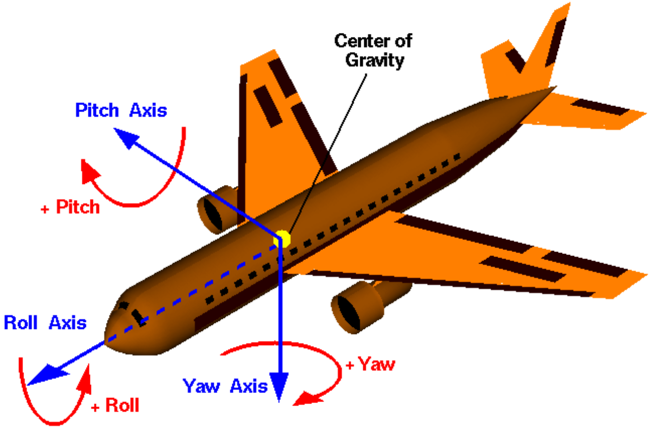
3D Rotations

- Advantage:
  Can be easily concatenated and inverted (how?)
- Disadvantage:
  Over-parameterized (9 parameters instead of 3)

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Euler Angles

- Product of 3 consecutive rotations (e.g., around X-Y-Z axes)
- Roll-pitch-yaw convention is very common in aerial navigation (DIN 9300)



http://en.wikipedia.org/wiki/File:Rollpitchyawplain.png

Roll-Pitch-Yaw Convention

- Roll ϕ, Pitch θ, Yaw ψ
- Conversion to 3x3 rotation matrix:

$$\mathbf{R} = \mathbf{R}_Z(\psi)\mathbf{R}_Y(\theta)\mathbf{R}_X(\phi)$$

$$= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

$$= \begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

Roll-Pitch-Yaw Convention

- Roll ϕ, Pitch θ, Yaw ψ
- Conversion from 3x3 rotation matrix:

$$\phi = \operatorname{Atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\psi = -\operatorname{Atan2}\left(\frac{r_{21}}{\cos(\phi)}, \frac{r_{11}}{\cos(\phi)}\right)$$

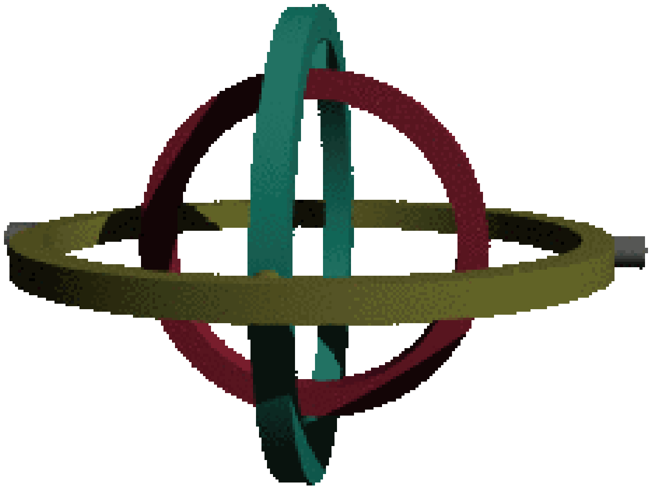
$$\theta = \operatorname{Atan2}\left(\frac{r_{32}}{\cos(\phi)}, \frac{r_{33}}{\cos(\phi)}\right)$$

Euler Angles

- Advantage:
  - Minimal representation (3 parameters)
  - Easy interpretation
- Disadvantages:
  - Many “alternative” Euler representations exist (XYZ, ZXZ, ZYX, …)
  - Difficult to concatenate
  - Singularities (gimbal lock)

Gimbal Lock

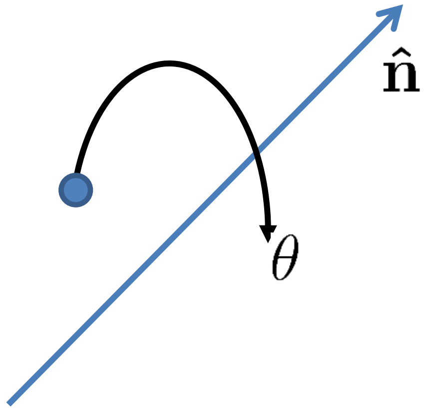
- When the axes align, one degree-of-freedom (DOF) is lost:



http://commons.wikimedia.org/wiki/File:Rotating_gimbal-xyz.gif

Axis/Angle

- Represent rotation by
  - rotation axis $\hat{\mathbf{n}}$ and
  - rotation angle θ
- 4 parameters
- 3 parameters
  - length is rotation angle
  - also called the angular velocity
  - minimal but not unique (why?)



Conversion

- Rodriguez' formula

$$\mathbf{R}(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)[\hat{\mathbf{n}}]_\times^2$$

- Inverse

$$\theta = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R}) - 1}{2}\right), \hat{\mathbf{n}} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

see: An Invitation to 3D Vision (Ma, Soatto, Kosecka, Sastry), Chapter 2

Axis/Angle

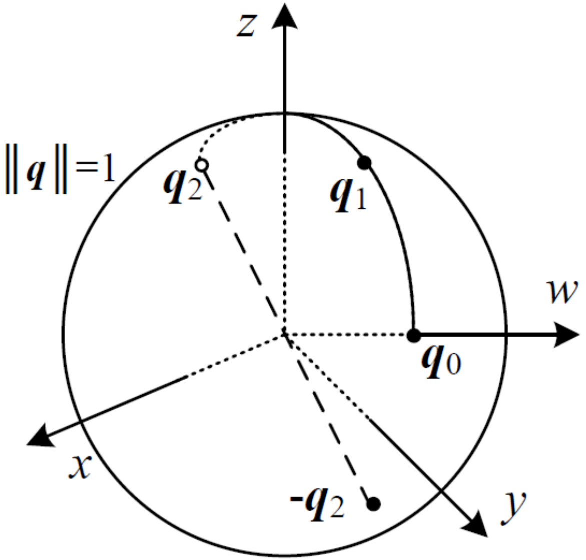
- Also called twist coordinates
- Advantages:
  - Minimal representation
  - Simple derivations
- Disadvantage:
  - Difficult to concatenate
  - Slow conversion

Quaternions

- Quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top \in \mathbb{R}^4$
- Real and vector part

$$\mathbf{q} = (r, \mathbf{v}),\ r \in \mathbb{R},\ \mathbf{v} \in \mathbb{R}^3$$

- Unit quaternions have $\|\mathbf{q}\| = 1$
- Opposite sign quaternions represent the same rotation
- Otherwise unique



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

Quaternions

- Advantage:
  multiplication, inversion and rotations are very efficient
- Concatenation

$$(r_1, \mathbf{v}_1)(r_2, \mathbf{v}_2) = (r_1 r_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, r_1 \mathbf{v}_2 + r_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

- Inverse (=flip signs of real or imaginary part)

$$(r, \mathbf{v})^{-1} = (r, \mathbf{v})^\star \equiv (-r, \mathbf{v}) \equiv (r, -\mathbf{v})$$

- Rotate 3D vector $\mathbf{p} \in \mathbb{R}^3$ using a quaternion:

$$(r, \mathbf{v})(0, \mathbf{p})(r, \mathbf{v})^\star$$

Quaternions

- Rotate 3D vector $\mathbf{p} \in \mathbb{R}^3$ using a quaternion:

$$(r, \mathbf{v})(0, \mathbf{p})(r, \mathbf{v})^\star$$

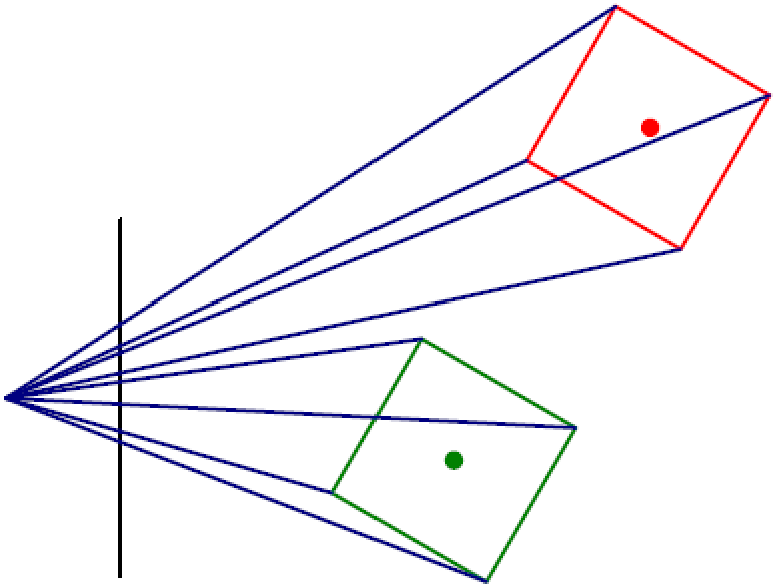
- Relation to axis/angle representation

$$\mathbf{q} = (r, \mathbf{v}) = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\mathbf{n}})$$

3D Orientations

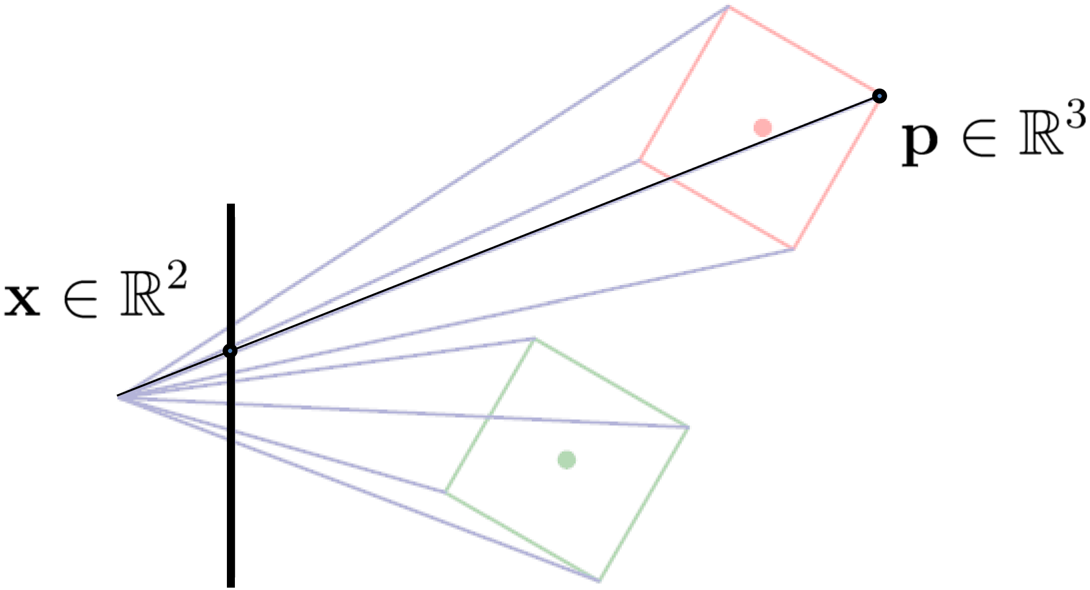
- **Note:** In general, it is very hard to "read" 3D orientations/rotations, no matter in what representation
- **Observation:** They are usually easy to visualize and can then be intuitively interpreted
- **Advice:** Use 3D visualization tools for debugging (RVIZ, libqglviewer, …)

3D to 2D Perspective Projections



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

3D to 2D Perspective Projections



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

3D to 2D Perspective Projections

- Pin-hole camera model

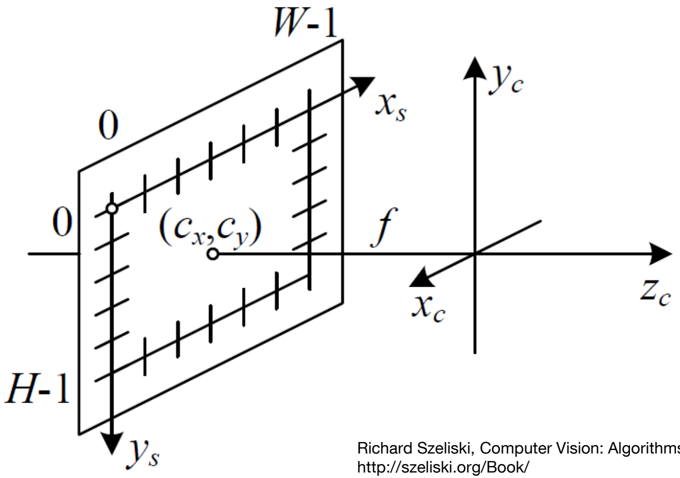
$$\tilde{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{\mathbf{p}}$$

- Note: $\tilde{\mathbf{x}}$ is homogeneous, needs to be normalized

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{pmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \end{pmatrix}$$

Camera Intrinsics

- So far, 2D point is given in meters on image plane
- But: we want 2D point be measured in pixels (as the sensor does)



Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

Camera Intrinsics

- Need to apply some scaling/offset

$$\tilde{\mathbf{x}} = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{intrinsics } K} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{projection}} \tilde{\mathbf{p}}$$

- Focal length f_x, f_y
- Camera center c_x, c_y
- Skew s

Camera Extrinsics

- Assume $\tilde{\mathbf{p}}_w$ is given in world coordinates
- Transform from world to camera (also called the camera extrinsics)

$$\tilde{\mathbf{p}} = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{p}}_w$$

- Projection of 3D world points to 2D pixel coordinates:

$$\tilde{\mathbf{x}} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & \mathbf{t} \end{pmatrix} \tilde{\mathbf{p}}_w$$

Lessons Learned

- 3D points, lines, planes
- 3D transformations
- Different representations for 3D orientations
  - Choice depends on application
  - Which representations do you remember?
- 3D to 2D perspective projections
- You **really** have to know 2D/3D transformations by heart (for more info, read Szeliski, Chapter 2, available online)